ПРОДЕЛАННАЯ РАБОТА ПО НИР НА ТЕМУ СИСТЕМА АВТОМАТИЧЕСКОГО ХАОС-ТЕСТИРОВАНИЯ ХОСТОВОГО АГЕНТА

4.1. Начало

Первым и основополагающим проделанным шагов является выбор темы НИР, научного руководителя и консультанта. Среди потенциальных объектов исследования выступали: penetration testing для хостового агента, chaos testing для хостового агента, статический анализ кода с точки зрения багов и потенциальных уязвимостей, система для распознавания коррелирующих событий в рамках SIEM, безопасное выполнение удаленных adhoc запросов на хостах. Тема хаос-тестирование хостового агента выделялась тем, что сохраняла баланс между интересной тематикой, приемлемому объему экспертизы, который предстояло набрать и запросить, а также актуальностью с точки зрения немедленного прикладного использования.

Научным руководителем был выбран Косицын Павел Андреевич, консультантом Кузнецов Александр. Они являются специалистами в своих областях: Павел часто имеет дело с распределенными высоконагруженными системами и знает много о специфичных методах тестирования; Александр занимается разработкой, в том числе хостовых агентов безопасности.

4.2. Актуальность темы

4.2.1. Что такое хаос-инжиниринг?

Хаос-инжиниринг (Chaos Engineering) — это практика в области разработки и тестирования программного обеспечения, направленная на повышение устойчивости и надежности сложных систем (почти всегда имеются в виду распределенные системы). Путем преднамеренного внедрения сбоев (fault injection) и непредвиденных ситуаций в контролируемой среде, хаос-инжиниринг позволяет выявлять уязвимости и недостатки в системе до того, как они проявятся в реальной эксплуатации и нанесут фатальный урон и компании, и пользователям.

Концепция хаос-инжиниринга стала обсуждаться в компании Netflix в конце 2000-х — начале 2010-х годов. В процессе перехода от монолитной архитектуры к микросервисам и миграции в облако, Netflix столкнулась с необходимостью обеспечения высокой доступности и устойчивости своих сервисов в условиях сложной распределенной системы. Чтобы проверить и улучшить устойчивость своей инфраструктуры, инженеры Netflix разработали инструмент под названием Chaos Monkey. Этот инструмент случайным образом отключал экземпляры служб в производственной среде, что вынуждало систему автоматически реагировать на сбои и перенаправлять трафик, обеспечивая непрерывность обслуживания пользователей. Успех Chaos Monkey привел к созданию целого набора инструментов, а также полноценных систем для хаос-тестирования.

4.2.2. Принципы хаос-инжиниринга

Хаос-инжиниринг основывается на нескольких ключевых принципах:

1. Гипотеза о стабильном состоянии: определение нормального состояния системы на основе ключевых метрик.

2. Внедрение сбоев: создание сценариев, максимально приближенных к реальным сбоям, которые могут произойти в системе.

3. Минимизация радиуса воздействия: проведение экспериментов в контролируемой среде с целью ограничения потенциального негативного влияния на пользователей.

4. Автоматизация экспериментов: использование инструментов для автоматического и повторяемого проведения тестов.

5. Мониторинг внешних метрик приложения, а не его внутренних проблем: тщательное наблюдение за метриками системы во время экспериментов, не принимая во внимания внутреннюю стратегию деградации и отказа.

4.2.3. Инструменты хаос-инжиниринга

Сегодня существует множество инструментов, облегчающих внедрение хаос-инжиниринга: Gremlin, Chaos Toolkit, LitmusChaos, Chaos Mesh, AWS

Fault Injection Simulator (FIS), Azure Chaos Studio, Pumba, PowerfulSeal. Некоторые из них разобраны в основном тексте работы с точки зрения применимости к хостовому агенту.

4.2.4. Почему только распределенные системы?

Довольно очевидна применимость хаос-инжиниринга в распределенных системах. Современные распределенные системы характеризуются высокой сложностью из-за микросервисной архитектуры, динамического масштабирования, распределения по зонам доступности. В таких условиях традиционные подходы к тестированию не могут дать гарантию надежности, а хаос-инжиниринг, наоборот, предлагает широкий диапазон сценариев для предотвращения финансовых и репутационных потерь при возникновении инцидентов.

Однако, поскольку акцент ставится именно на распределенных системах, все сценарии уходят “вширь„. Популярные инструменты для хаос-инжиниринга берут во внимание огромное количество компонент инфраструктуры, в которых может произойти отказ, что делает карту экспериментов широкой и исчерпывающий для сложно спроектированных систем. Когда речь заходит о небольших приложениях, то есть тривиальных с точки зрения распределенных систем, такой подход становится не столь применимым.

4.2.5. Тестирование хостовых агентов

Хостовыми агентами являются приложения, эксплуатация которых подразумевает функционирование на виртуальных или железных машинах, к которым у разработчиков нет прямого доступа. В современном мире хостовые агенты становятся неотъемлемой частью информационных систем, обеспечивая мониторинг, управление и безопасность на удалённых устройствах и серверах. Они выполняют критически важные функции, такие как сбор данных, анализ производительности, обнаружение вторжений и реагирование на инциденты.

Но в рамках такой стратегии использования небольшие сервисы становятся не менее “опасными„ чем сложные системы. Поскольку агенты функционируют в разнообразных и иногда тяжело предсказуемых средах, требуются надёжные методы тестирования, гарантирующие их корректную работу

в условиях различных сбоев и аномалий.

Получается, тут напрашивается тот же подход хаос-инжиниринга, но смотреть нужно уже “вглубь„.

Хостовые агенты взаимодействуют с меньшим числом компонент, им не нужны широкие эксперименты, но нужны тонко настроенные, учитывающие специфику окружения, вплоть до версии OS (Operating system) и SO(Shared objects). Использовать популярные инструменты хаос-тестирования для этого намного сложнее, к тому же возможность покрыть специфичные сценарии вызывает вопросы. Именно этим обеспечивается актуальность и новизна заявленной темы НИР.

4.2.6. Чего боится хостовой агент?

Следующим шагом в работе было определение ключевых требований к системе автоматического хаос-тестирования хостового агента. Функциональные требования напрямую вытекали из критических точек отказа, о которых разработчики, как правило, не имеют возможности побеспокоиться, оперируя классическими методами тестирования. К таким точкам относятся оперативная память, CPU, диск, сеть, а также смежные относительно приложения утилиты, с которыми агент плотно взаимодействует. В каждой из точек потенциального сбоя были детально описаны губительные для агента процессы.

Сформулированные нефункциональные требования обусловлены принципами хаос-инжиниринга и потребностью в воспроизводимости экспериментов.

4.2.7. Что можно сделать уже сейчас?

Следующим шагов в работе было оценить применимость и доступность существующих инструментов хаос-тестирования для хостового агента. Были сделаны выводы о том, почему большинство инструментов не могут быть использованы для достижения целей. Чаще всего причиной этому является их работа “широкими мазками„: они не могут отключить или сломать что-то маленькое, их неделимая единица слишком велика для приложения, функционирующего на единственном поде.

Самым многообещающим показал себя сервис Chaos Mesh. В его арсена-

ле есть инструменты для внедрения сбоев в работу сети, процессора, оперативной памяти, операций ввода/вывода на диск. Важным преимуществом сервиса является возможность проведения экспериментов, как и на виртуальных машинах, так и в окружении Kubernetes. Именно с Chaos Mesh будут происходит дальнейшие манипуляции в НИР: ему предстоит проверки на возможность конфигурирования специфичных сценариев и интеграции с дополнительными утилитами, в том числе для более детального взгляда на происходящее внутри хостового приложения во время экспериментов. Последнее является практически отступлением от одного из принципов хаос-инжиниринга: вместо метрик приложения, акцент будет сделан на покрытии кода и веток.